



US009158628B2

(12) **United States Patent**
Maity et al.

(10) **Patent No.:** **US 9,158,628 B2**
(45) **Date of Patent:** **Oct. 13, 2015**

(54) **BIOS FAILOVER UPDATE WITH SERVICE PROCESSOR HAVING DIRECT SERIAL PERIPHERAL INTERFACE (SPI) ACCESS**

G06F 11/1417; G06F 11/1433; G06F 11/20;
G06F 11/073; G06F 11/076
See application file for complete search history.

(71) Applicant: **AMERICAN MEGATRENDS, INC.,**
Norcross, GA (US)

(56) **References Cited**

U.S. PATENT DOCUMENTS

(72) Inventors: **Sanjoy Maity**, Snellville, GA (US);
Baskar Parthiban, Johns Creek, GA (US);
Satheesh Thomas, Norcross, GA (US);
Purandhar Nallagatla, Johns Creek, GA (US);
Ramakoti Reddy Bhimanadhuni, Suwanee, GA (US);
Harikrishna Doppalapudi, Norcross, GA (US)

6,944,758	B2 *	9/2005	Lin	713/2
7,043,666	B2 *	5/2006	Wynn et al.	714/6.13
7,765,393	B1 *	7/2010	Lee et al.	713/2
2003/0076311	A1 *	4/2003	Lin et al.	345/204
2004/0153738	A1 *	8/2004	Otake et al.	714/6
2004/0193862	A1 *	9/2004	Lin	713/1
2015/0081829	A1 *	3/2015	Maity et al.	709/212
2015/0143095	A1 *	5/2015	Maity et al.	714/15
2015/0149750	A1 *	5/2015	Maity et al.	714/15

* cited by examiner

(73) Assignee: **AMERICAN MEGATRENDS, INC.,**
Norcross, GA (US)

Primary Examiner — Nadeem Iqbal

(*) Notice: Subject to any disclaimer, the term of this patent is extended or adjusted under 35 U.S.C. 154(b) by 146 days.

(74) *Attorney, Agent, or Firm* — Locke Lord LLP; Tim Tingkang Xia, Esq.

(57)

ABSTRACT

Certain aspects direct to systems and methods of BIOS failover update with a service processor (SP) having direct serial peripheral interface (SPI) access to a basic input/output system (BIOS) chip of a host computer. In certain embodiments, the SP receives a failover backup image from a BIOS being executed at a CPU of the host computer through a system interface, and stores the failover backup image in the volatile memory. Then the SP monitors operation of the BIOS by receiving, from the BIOS, a notification signal. When the SP detects an error in the operation of the BIOS based on the notification signal, the SP sends a copy of the failover backup image to the BIOS chip of the host computer through the SPI to replace a current BIOS image stored in the BIOS chip of the host computer with the copy of the failover backup image.

(21) Appl. No.: **14/092,489**

(22) Filed: **Nov. 27, 2013**

(65) **Prior Publication Data**

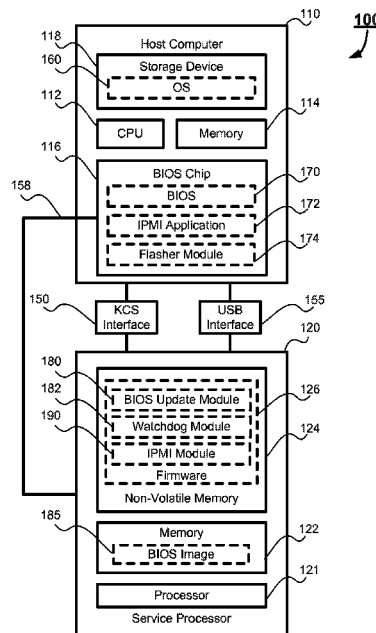
US 2015/0149815 A1 May 28, 2015

(51) **Int. Cl.**
G06F 11/14 (2006.01)

(52) **U.S. Cl.**
CPC **G06F 11/1435** (2013.01)

(58) **Field of Classification Search**
CPC G06F 11/1435; G06F 11/1451; G06F 11/1471; G06F 11/1464; G06F 11/1666;

29 Claims, 6 Drawing Sheets



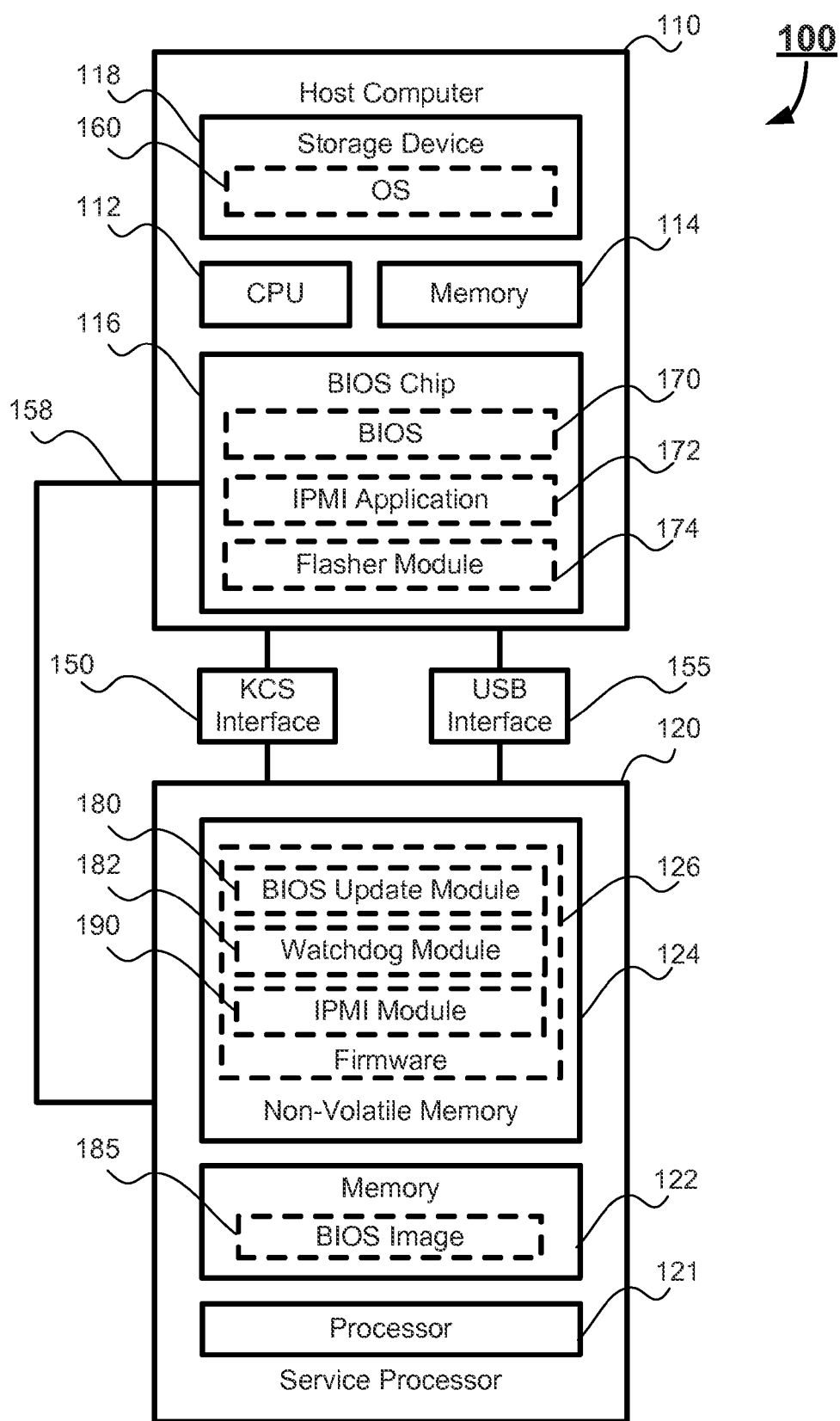


FIG. 1

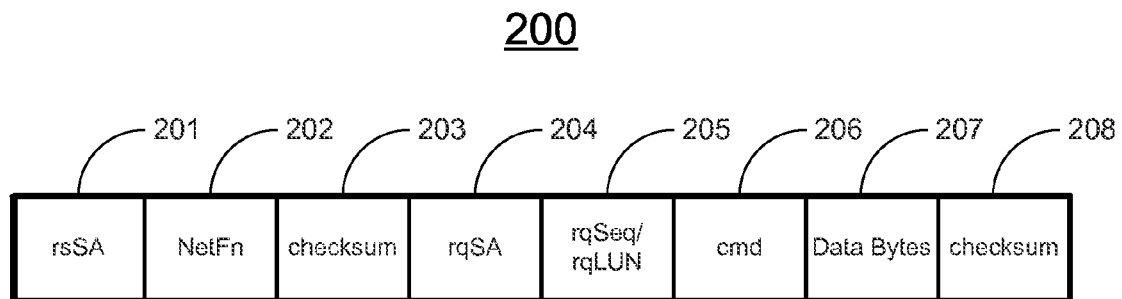


FIG. 2

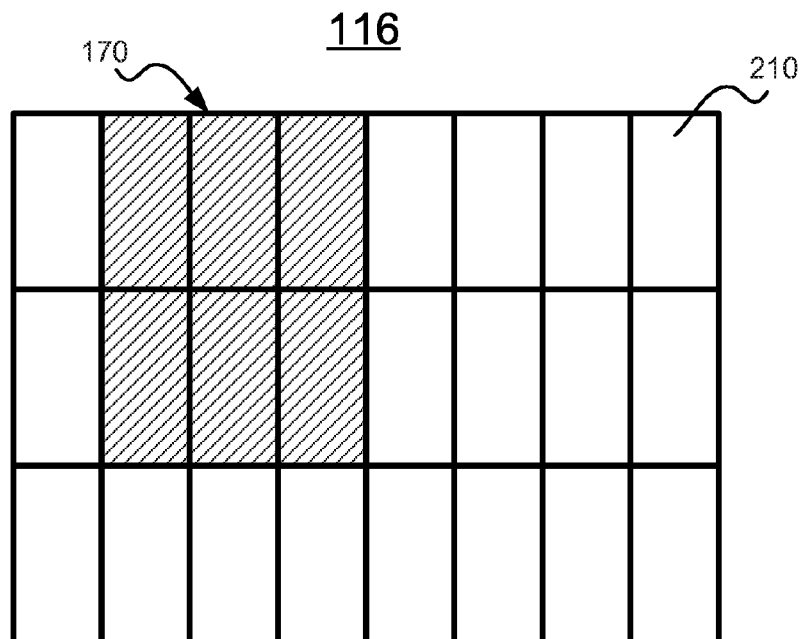


FIG. 3

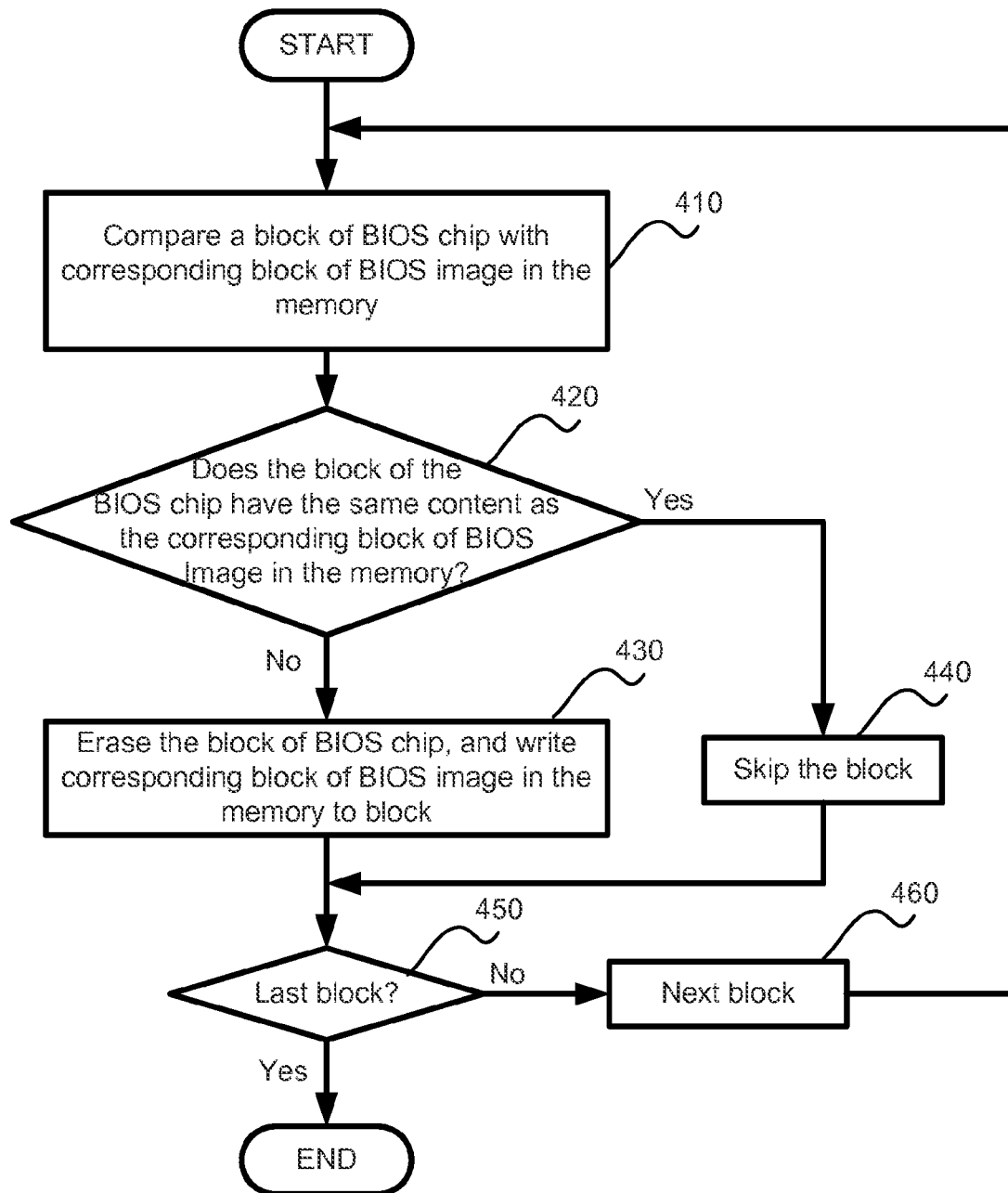


FIG. 4

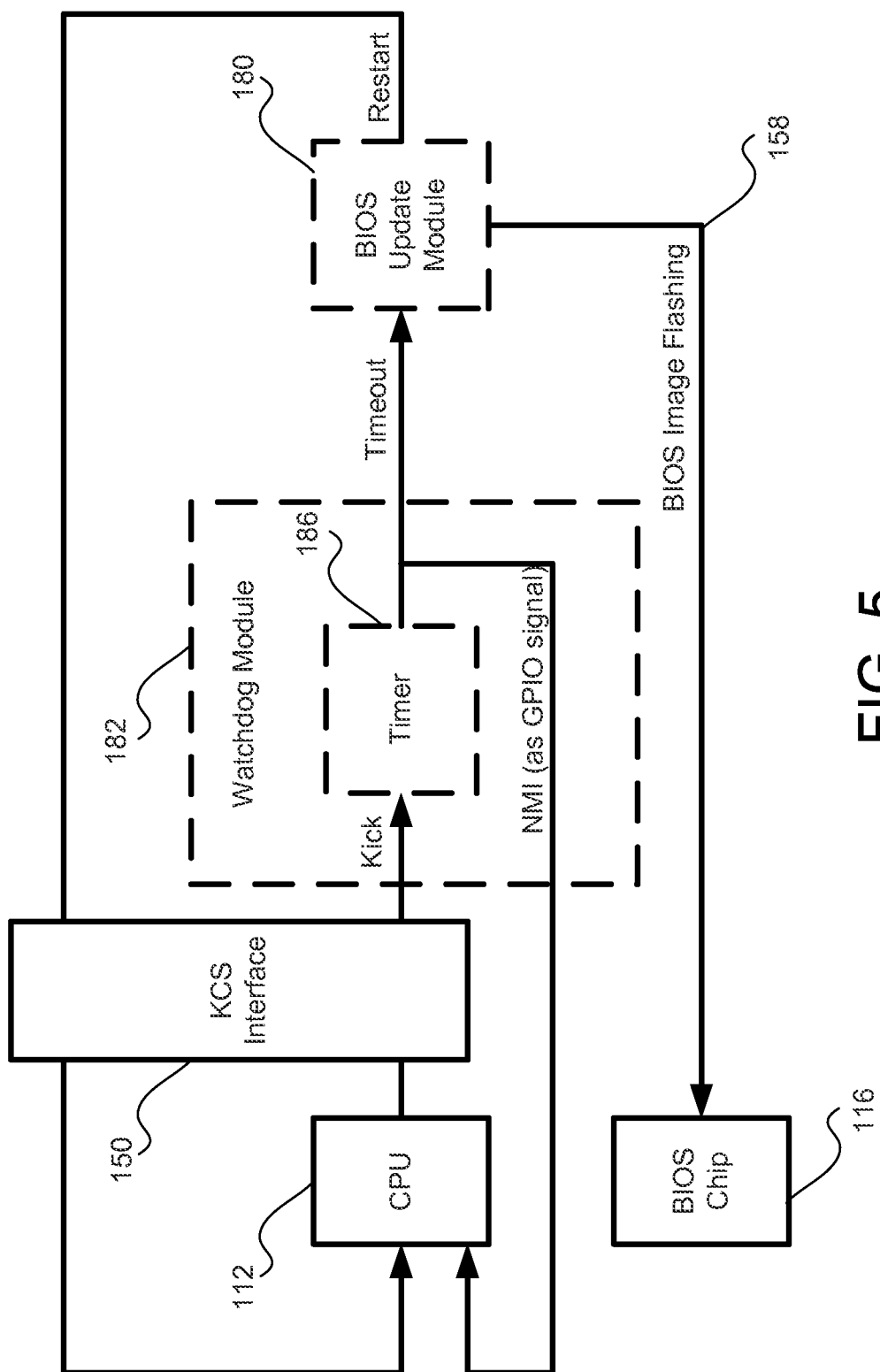


FIG. 5

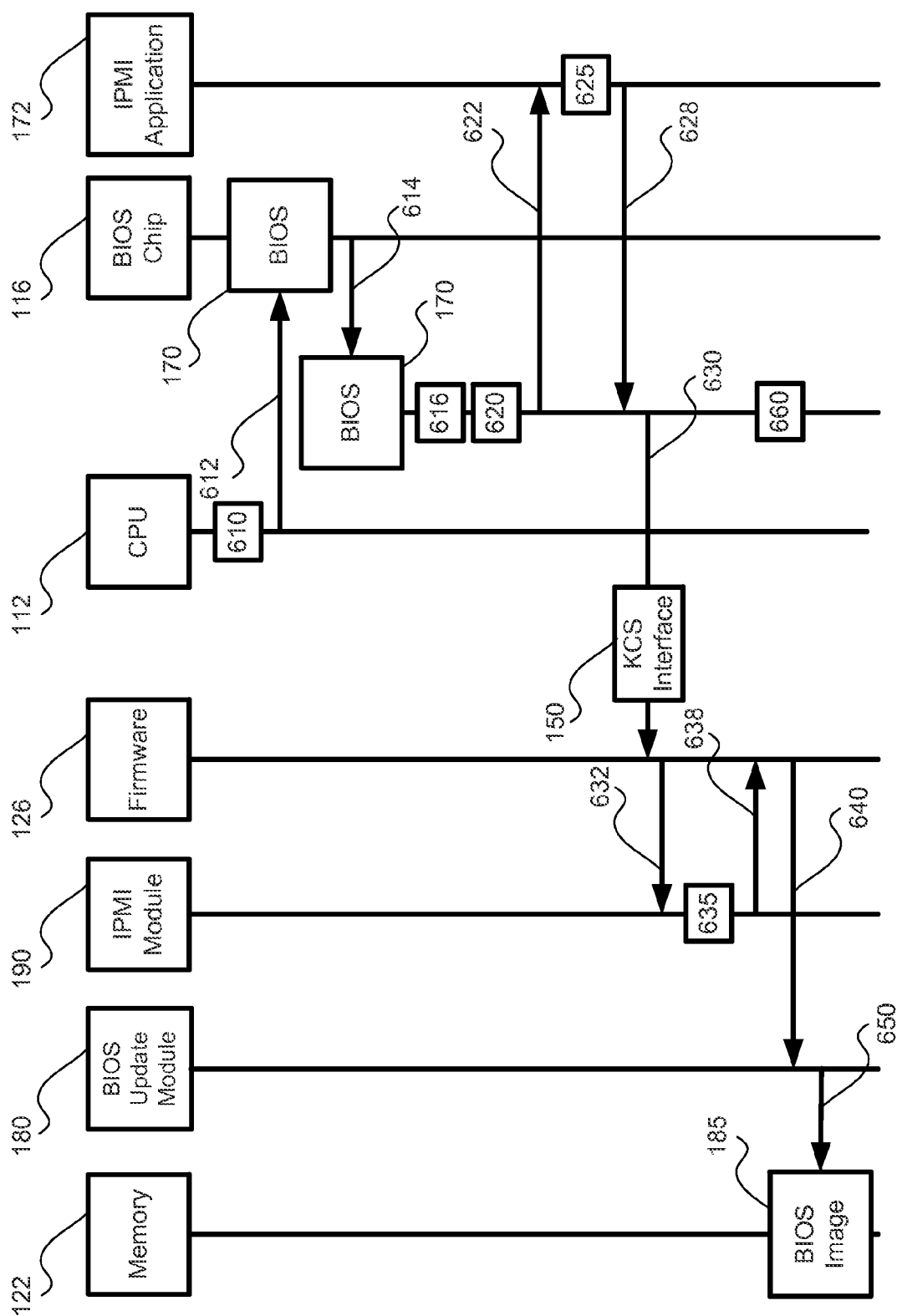


FIG. 6

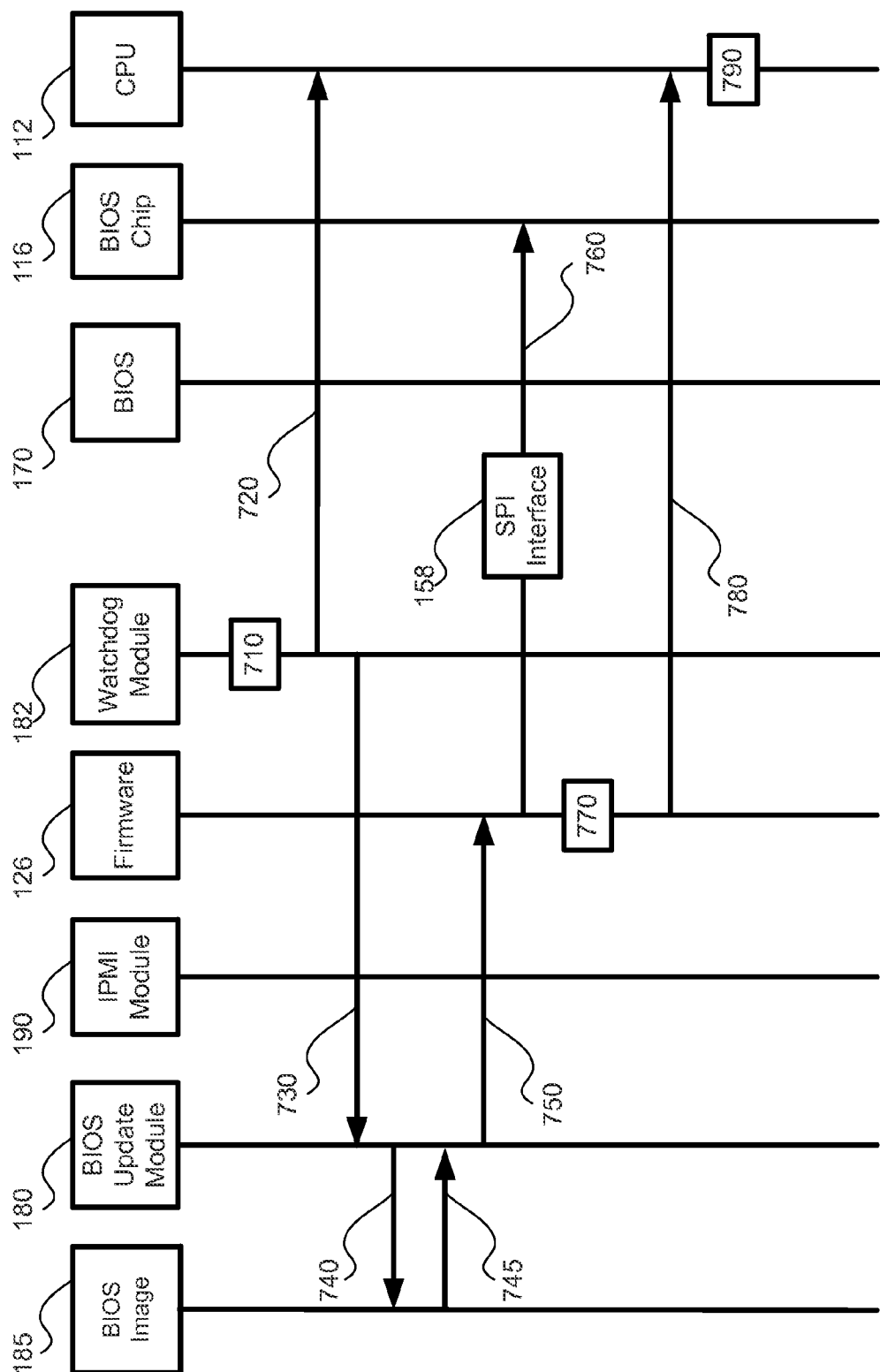


FIG. 7

1

BIOS FAILOVER UPDATE WITH SERVICE PROCESSOR HAVING DIRECT SERIAL PERIPHERAL INTERFACE (SPI) ACCESS

FIELD

The present disclosure relates generally to basic input/output system (BIOS) update of a host computer with a service processor (SP), and particularly to BIOS failover update with a SP having direct serial peripheral interface (SPI) access to the BIOS chip of the host computer.

BACKGROUND

The background description provided herein is for the purpose of generally presenting the context of the disclosure. Work of the presently named inventors, to the extent it is described in this background section, as well as aspects of the description that may not otherwise qualify as prior art at the time of filing, are neither expressly nor impliedly admitted as prior art against the present disclosure.

The Basic Input/Output System (BIOS) is one of the most crucial components on a computer motherboard. The BIOS firmware is preloaded into a memory (the BIOS memory) of the BIOS, and typically is the first code run by a computer when powered on. When the computer starts up, the first job for the BIOS is the power-on self-test, which initializes and identifies the system devices such as the CPU, RAM, video display card, keyboard and mouse, hard disk drive, optical disc drive and other hardware. The BIOS then locates a boot loader software held on a peripheral device (designated as a "boot device"), such as a hard disk or a CD/DVD, and loads and executes that software, giving it control of the operating system (OS). This process is known as booting, or booting up, which is short for bootstrapping.

The BIOS memory for storing the BIOS firmware is generally a non-volatile chip, such as an EEPROM chip on the motherboard. Typically, a serial peripheral interface (SPI) bus is used for the EEPROM chip. In modern computer systems, contents stored in the BIOS chip can be rewritten without removing it from the motherboard, allowing the BIOS firmware to be upgraded in place. The rewriting process of the BIOS firmware is generally referred to as flashing the BIOS.

Generally, a service processor (SP) or a baseboard management controller (BMC) refers to a specialized microcontroller that manages the interface between system management software and platform hardware. The SP can be embedded on the motherboard of a computer, generally a server. For example, different types of sensors can be built into the computer system, and the SP reads these sensors to obtain parameters such as temperature, cooling fan speeds, power status, operating system (OS) status, etc. The SP monitors the sensors and can send alerts to a system administrator via the network if any of the parameters do not stay within preset limits, indicating a potential failure of the system. The administrator can also remotely communicate with the BMC to take some corrective action such as resetting or power cycling the system to get a hung OS running again. The SP also provides out-of-band (OOB) access to the BIOS.

In a host computer provided with a SP, the SP can be used for flashing the BIOS. Generally, the SP may control the host computer to enter a special state to write the BIOS firmware image to the BIOS chip through the BIOS serial peripheral interface (SPI). However, in case of corruption to the BIOS image in the BIOS chip, the BIOS may fail during its execu-

2

tion, and the booting of the OS may not happen. There is a need to update the BIOS firmware image with the SP when such corruption occurs.

Therefore, an unaddressed need exists in the art to address the aforementioned deficiencies and inadequacies.

SUMMARY

Certain aspects of the present disclosure direct to a system. In certain embodiments, the system includes a service processor (SP), which includes a processor, a non-volatile memory, a volatile memory and a system interface. The non-volatile memory stores computer executable codes are configured to, when executed at the processor: receive, from a basic input/output system (BIOS) being executed at a central processing unit (CPU) of a host computer through a system interface, a BIOS image as a failover backup image, and store the failover backup image in the volatile memory, wherein the SP is connected to the host computer via the system interface, and wherein the SP is connected to a BIOS chip of the host computer via a serial peripheral interface (SPI); monitor operation of the BIOS being executed at the CPU of the host computer by receiving, from the BIOS through the system interface, a notification signal; detect, based on the notification signal, an error in the operation of the BIOS being executed at the CPU of the host computer; and in response to detecting the error, send a copy of the failover backup image to the BIOS chip of the host computer through the SPI to replace a current BIOS image stored in the BIOS chip of the host computer with the copy of the failover backup image.

In certain embodiments, the system interface is a standardized interface under an Intelligent Platform Management Interface (IPMI) standard, wherein the standardized interface includes a keyboard controller style (KCS) interface, a system management interface chip (SMIC) interface, and a block transfer (BT) interface.

In certain embodiments, the system interface is a universal serial bus (USB) interface.

In certain embodiments, the SP is a baseboard management controller (BMC).

In certain embodiments, the codes include: a detection module configured to receive the notification signal from the BIOS being executed at the CPU of the host computer through the system interface, and to detect the error based on the notification signal. In certain embodiments, the detection module is configured to detect the error when the detection module does not receive the notification signal for a predetermined period of time.

In certain embodiments, the host computer includes: the BIOS chip storing the current BIOS image; a memory; and the CPU, configured to load a copy of the current BIOS image to the memory and execute the current BIOS image as the BIOS. The BIOS, when executed at the CPU, is configured to: retrieve a copy of the current BIOS image from the BIOS chip, and send the copy of the current BIOS image as the failover backup image to the SP through the system interface; and send the notification signal to the SP during the operation of the BIOS.

In certain embodiments, the BIOS, when executed at the CPU, is configured to send the copy of the current BIOS image as the failover backup image to the SP by: generating a first IPMI original equipment manufacturer (OEM) message comprising the copy of the current BIOS image; and sending the first IPMI OEM message to the SP through the system interface.

In certain embodiments, the codes are configured to receive the BIOS image as the failover backup image from the

BIOS being executed at the CPU of the host computer by: receiving the first IPMI OEM message from the BIOS being executed at the CPU of the host computer through the system interface; and retrieving the BIOS image from the first IPMI OEM message.

In certain embodiments, the notification signal is a second IPMI OEM message.

In certain embodiments, the BIOS, when executed at the CPU, is further configured to: generate a request for version information of the failover backup image, and send the request for the version information of the failover backup image to the SP through the system interface; receive, from the SP through the system interface, the version information of the failover backup image; and compare the version information of the failover backup image to a version information of the current BIOS image.

In certain embodiments, the codes are configured to: receive, from the BIOS being executed at the CPU of the host computer through the system interface, the request for the version information of the failover backup image; in response to the request for the version information, retrieve the version information from the failover backup image stored in the volatile memory; and send the version information to the host computer through the system interface.

In certain embodiments, the request for the version information of the failover backup image is a third IPMI OEM message, and the version information of the failover backup image is a fourth IPMI OEM message.

Certain aspects of the present disclosure direct to a method of failover updating a basic input/output system (BIOS) of a host computer with a service processor (SP), including: receiving, at the SP, a failover backup image from the host computer through a system interface, and storing the failover backup image in a volatile memory of the SP, wherein the host computer includes a central processing unit (CPU), a memory and a BIOS chip storing a current BIOS image, wherein the CPU is configured to load a copy of the current BIOS image to the memory and execute the current BIOS image as a BIOS, wherein the BIOS being executed at the CPU of the host computer is configured to retrieve a copy of the current BIOS image from the BIOS chip and send the copy of the current BIOS image as the failover backup image to the SP through the system interface, and wherein the SP is connected to the BIOS chip of the host computer via a serial peripheral interface (SPI); monitoring, at the SP, operation of the BIOS being executed at the CPU of the host computer by receiving, from the BIOS through the system interface, a notification signal; detecting, at the SP, an error in the operation of the BIOS being executed at the CPU of the host computer based on the notification signal; and in response to detecting the error, sending, by the SP, a copy of the failover backup image to the BIOS chip of the host computer through the SPI to replace the current BIOS image stored in the BIOS chip of the host computer with the copy of the failover backup image.

In certain embodiments, the system interface is a standardized interface under an Intelligent Platform Management Interface (IPMI) standard, wherein the standardized interface includes a keyboard controller style (KCS) interface, a system management interface chip (SMIC) interface, and a block transfer (BT) interface.

In certain embodiments, the system interface is a universal serial bus (USB) interface.

In certain embodiments, the SP is a baseboard management controller (BMC).

In certain embodiments, the SP includes a detection module configured to receive the notification signal from the

BIOS being executed at the CPU of the host computer through the system interface, and to detect the error based on the notification signal.

In certain embodiments, the detection module is configured to detect the error when the detection module does not receive the notification signal for a predetermined period of time.

In certain embodiments, the SP is configured to receive the failover backup image from the BIOS being executed at the CPU of the host computer by: receiving a first IPMI original equipment manufacturer (OEM) message from the BIOS being executed at the CPU of the host computer through the system interface, wherein the first IPMI OEM message is generated by the BIOS being executed at the CPU of the host computer and comprises the copy of the current BIOS image; and retrieving the copy of the current BIOS image as the failover backup image from the first IPMI OEM message.

In certain embodiments, the notification signal is a second IPMI OEM message.

In certain embodiments, the method further includes: receiving, at the SP, a request for the version information of the failover backup image from the BIOS being executed at the CPU of the host computer through the system interface; in response to the request for the version information, retrieving, at the SP, the version information from the failover backup image stored in the volatile memory; and sending, from the SP, the version information to the host computer through the system interface, wherein the the BIOS being executed at the CPU of the host computer receives and compares the version information of the failover backup image to a version information of the current BIOS image. In certain embodiments, the request for the version information of the failover backup image is a third IPMI OEM message, and the version information of the failover backup image is a fourth IPMI OEM message.

Certain aspects of the present disclosure direct to a non-transitory computer readable medium storing computer executable codes. The codes, when executed at a processor of a service processor (SP), are configured to: receive a failover backup image from a host computer through a system interface, and storing the failover backup image in a volatile memory of the SP, wherein the host computer comprises a central processing unit (CPU), a memory and a BIOS chip storing a current BIOS image, wherein the CPU is configured to load a copy of the current BIOS image to the memory and execute the current BIOS image as a BIOS, wherein the BIOS being executed at the CPU of the host computer is configured to retrieve a copy of the current BIOS image from the BIOS chip and send the copy of the current BIOS image as the failover backup image to the SP through the system interface, and wherein the SP is connected to the BIOS chip of the host computer via a serial peripheral interface (SPI); monitor operation of the BIOS being executed at the CPU of the host computer by receiving, from the BIOS through the system interface, a notification signal; detect an error in the operation of the BIOS being executed at the CPU of the host computer based on the notification signal; and in response to detecting the error, send a copy of the failover backup image to the BIOS chip of the host computer through the SPI to replace the current BIOS image stored in the BIOS chip of the host computer with the copy of the failover backup image.

In certain embodiments, the system interface is a standardized interface under an Intelligent Platform Management Interface (IPMI) standard, wherein the standardized interface includes a keyboard controller style (KCS) interface, a system management interface chip (SMIC) interface, and a block transfer (BT) interface.

5

In certain embodiments, the system interface is a universal serial bus (USB) interface.

In certain embodiments, the SP is a baseboard management controller (BMC).

In certain embodiments, the codes includes a detection module configured to receive the notification signal from the BIOS being executed at the CPU of the host computer through the system interface, and to detect the error based on the notification signal. In certain embodiments, the detection module is configured to detect the error when the detection module does not receive the notification signal for a prede-
termined period of time.

In certain embodiments, the codes are configured to receive the failover backup image from the BIOS being executed at the CPU of the host computer by: receiving a first IPMI original equipment manufacturer (OEM) message from the BIOS being executed at the CPU of the host computer through the system interface, wherein the first IPMI OEM message is generated by the BIOS being executed at the CPU of the host computer and comprises the copy of the current BIOS image; and retrieving the copy of the current BIOS image as the failover backup image from the first IPMI OEM message.

In certain embodiments, the notification signal is a second IPMI OEM message.

In certain embodiments, the codes are further configured to: receive a request for the version information of the failover backup image from the BIOS being executed at the CPU of the host computer through the system interface; in response to the request for the version information, retrieve the version information from the failover backup image stored in the volatile memory; and send the version information to the host computer through the system interface, wherein the BIOS being executed at the CPU of the host computer receives and compares the version information of the failover backup image to a version information of the current BIOS image. In certain embodiments, the request for the version information of the failover backup image is a third IPMI OEM message, and the version information of the failover backup image is a fourth IPMI OEM message.

These and other aspects of the present disclosure will become apparent from the following description of the preferred embodiment taken in conjunction with the following drawings and their captions, although variations and modifications therein may be affected without departing from the spirit and scope of the novel concepts of the disclosure.

BRIEF DESCRIPTION OF THE DRAWINGS

The present disclosure will become more fully understood from the detailed description and the accompanying drawings, wherein:

FIG. 1 schematically depicts a computer system according to certain embodiments of the present disclosure;

FIG. 2 schematically depicts an IPMI message according to certain embodiments of the present disclosure;

FIG. 3 schematically depicts the BIOS chip storing the BIOS according to certain embodiments of the present disclosure;

FIG. 4 schematically depicts a flowchart of a flashing process of the BIOS chip according to certain embodiments of the present disclosure;

FIG. 5 schematically depicts operation of a watchdog module according to certain embodiments of the present disclosure;

6

FIG. 6 schematically depicts a SP receiving a failover backup BIOS image from the BIOS of the host computer in the booting process according to certain embodiments of the present disclosure; and

FIG. 7 schematically depicts a BIOS failover flashing process initiated by the SP when the SP has direct SPI access to the BIOS chip according to certain embodiments of the present disclosure.

DETAILED DESCRIPTION

The present disclosure is more particularly described in the following examples that are intended as illustrative only since numerous modifications and variations therein will be apparent to those skilled in the art. Various embodiments of the disclosure are now described in detail. Referring to the drawings, like numbers, if any, indicate like components throughout the views. As used in the description herein and throughout the claims that follow, the meaning of “a”, “an”, and “the” includes plural reference unless the context clearly dictates otherwise. Also, as used in the description herein and throughout the claims that follow, the meaning of “in” includes “in” and “on” unless the context clearly dictates otherwise. Moreover, titles or subtitles may be used in the specification for the convenience of a reader, which shall have no influence on the scope of the present disclosure. Additionally, some terms used in this specification are more specifically defined below.

The terms used in this specification generally have their ordinary meanings in the art, within the context of the disclosure, and in the specific context where each term is used. Certain terms that are used to describe the disclosure are discussed below, or elsewhere in the specification, to provide additional guidance to the practitioner regarding the description of the disclosure. For convenience, certain terms may be highlighted, for example using italics and/or quotation marks. The use of highlighting has no influence on the scope and meaning of a term; the scope and meaning of a term is the same, in the same context, whether or not it is highlighted. It will be appreciated that same thing can be said in more than one way. Consequently, alternative language and synonyms may be used for any one or more of the terms discussed herein, nor is any special significance to be placed upon whether or not a term is elaborated or discussed herein. Synonyms for certain terms are provided. A recital of one or more synonyms does not exclude the use of other synonyms. The use of examples anywhere in this specification including examples of any terms discussed herein is illustrative only, and in no way limits the scope and meaning of the disclosure or of any exemplified term. Likewise, the disclosure is not limited to various embodiments given in this specification.

Unless otherwise defined, all technical and scientific terms used herein have the same meaning as commonly understood by one of ordinary skill in the art to which this disclosure pertains. In the case of conflict, the present document, including definitions will control.

As used herein, “around”, “about” or “approximately” shall generally mean within 20 percent, preferably within 10 percent, and more preferably within 5 percent of a given value or range. Numerical quantities given herein are approximate, meaning that the term “around”, “about” or “approximately” can be inferred if not expressly stated.

As used herein, “plurality” means two or more.

As used herein, the terms “comprising,” “including,” “carrying,” “having,” “containing,” “involving,” and the like are to be understood to be open-ended, i.e., to mean including but not limited to.

As used herein, the phrase at least one of A, B, and C should be construed to mean a logical (A or B or C), using a non-exclusive logical OR. It should be understood that one or more steps within a method may be executed in different order (or concurrently) without altering the principles of the present disclosure.

As used herein, the term “module” may refer to, be part of, or include an Application Specific Integrated Circuit (ASIC); an electronic circuit; a combinational logic circuit; a field programmable gate array (FPGA); a processor (shared, dedicated, or group) that executes code; other suitable hardware components that provide the described functionality; or a combination of some or all of the above, such as in a system-on-chip. The term module may include memory (shared, dedicated, or group) that stores code executed by the processor.

The term “code”, as used herein, may include software, firmware, and/or microcode, and may refer to programs, routines, functions, classes, and/or objects. The term shared, as used above, means that some or all code from multiple modules may be executed using a single (shared) processor. In addition, some or all code from multiple modules may be stored by a single (shared) memory. The term group, as used above, means that some or all code from a single module may be executed using a group of processors. In addition, some or all code from a single module may be stored using a group of memories.

As used herein, the term “headless system” or “headless machine” generally refers to the computer system or machine that has been configured to operate without a monitor (the missing “head”), keyboard, and mouse.

The term “interface”, as used herein, generally refers to a communication tool or means at a point of interaction between components for performing data communication between the components. Generally, an interface may be applicable at the level of both hardware and software, and may be uni-directional or bi-directional interface. Examples of physical hardware interface may include electrical connectors, buses, ports, cables, terminals, and other I/O devices or components. The components in communication with the interface may be, for example, multiple components or peripheral devices of a computer system.

The terms “chip” or “computer chip”, as used herein, generally refer to a hardware electronic component, and may refer to or include a small electronic circuit unit, also known as an integrated circuit (IC), or a combination of electronic circuits or ICs.

The present disclosure relates to computer systems. As depicted in the drawings, computer components may include physical hardware components, which are shown as solid line blocks, and virtual software components, which are shown as dashed line blocks. One of ordinary skill in the art would appreciate that, unless otherwise indicated, these computer components may be implemented in, but not limited to, the forms of software, firmware or hardware components, or a combination thereof.

The apparatuses and methods described herein may be implemented by one or more computer programs executed by one or more processors. The computer programs include processor-executable instructions that are stored on a non-transitory tangible computer readable medium. The computer programs may also include stored data. Non-limiting examples of the non-transitory tangible computer readable medium are nonvolatile memory, magnetic storage, and optical storage.

FIG. 1 schematically depicts a computer system according to certain embodiments of the present disclosure. As shown in

FIG. 1, the computer system **100** includes a host computer **110** and a service processor (SP) **120**. The SP **120** is connected to the host computer **110** via a keyboard controller style (KCS) interface **150** and a universal serial bus (USB) interface **155**. In certain embodiments, a remote management computer can be connected to the SP **120** via a network. The system **100** can be a system that incorporates more than one interconnected system, such as a client-server network. The network may be a wired or wireless network, and may be of various forms such as a local area network (LAN) or wide area network (WAN) including the Internet. In certain embodiments, the system **100** may include other physical or virtual components not shown in FIG. 1.

In certain embodiments, the SP **120** may be connected to the host computer **110** via only one of the KCS interface **150** and the USB interface **155**. For example, the SP **120** may be connected to the host computer **110** via only the KCS interface **150**, or via only the USB interface **155**.

In certain embodiments, the SP **120** may be connected to the host computer **110** via one or more interfaces replacing or in addition to the KCS interface **150** and the USB interface **155**. For example, the BMC **120** may be connected to the host computer **110** via other typical standardized Intelligent Platform Management Interface (IPMI) system interfaces, such as a system management interface chip (SMIC) interface or a block transfer (BT) interface. In certain embodiments, the SMIC interface and/or the BT interface can be used to replace one or both of the KCS interface **150** and the USB interface **155**. In certain embodiments, data transfer between the host computer **110** and the SP **120** can be in the format of IPMI original equipment manufacturer (OEM) messages, and goes through one of the KCS interface **150** and the USB interface **155**.

The host computer **110** may be a computing device, such as a general purpose computer or a headless computer. Generally, the host computer **110** includes a baseboard or the “motherboard” (not shown). The baseboard is a printed circuit board to which a multitude of components or devices may be connected by way of a system bus or other electrical communication paths. Although not explicitly shown in FIG. 1, the components on the baseboard are interconnected, and the layout of the components on the baseboard and the manner of the interconnection between the components on the baseboard is herein referred to as the configuration of the baseboard. One of ordinary skill in the art would appreciate that the configuration of the baseboard may be adjusted or changed according to the necessary design or manufacturing requirements.

The components on the baseboard of the host computer **110** include, but not limited to, a central processing unit (CPU) **112**, a memory **114**, a BIOS chip **116**, and other required memory and Input/Output (I/O) modules (not shown). In certain embodiments, the SP **120** may also be a component on the baseboard. In certain embodiments, the CPU **112**, the memory **114**, and the BIOS chip **116** may be embedded on the baseboard, or may be connected to the baseboard through at least one interface. In certain embodiments, the interface may be physical hardware interface such as electrical connectors, buses, ports, cables, terminals, or other I/O devices.

Further, the host computer **110** includes a storage device **118**, which stores a plurality of software applications, including an operating system (OS) **160**. In certain embodiments, the host computer **110** may include at least one I/O device (not shown) for generating and controlling input and output signals of the host computer **110**. Examples of the I/O device include keyboards, touch pads, mouse, microphones, display

screens, touch screens, or other I/O devices applicable for the host computer 110. Some I/O devices, such as touch screens, are provided for the dual input/output purposes.

The CPU 112 is a host processor which is configured to control operation of the host computer 110. The host processor can execute the OS 160 or other applications of the host computer 110. In certain embodiments, the host computer 110 may run on or more than one CPU as the host processor, such as two CPUs, four CPUs, eight CPUs, or any suitable number of CPUs.

In certain embodiments, the CPU 112 is configured to receive an interrupt, which is an asynchronous signal indicating the need for attention or a synchronous event in software indicating the need for a change in execution. Interrupts are a commonly used technique for computer multitasking, especially in real-time computing. Generally, an interrupt can be either a hardware interrupt or a software interrupt. The hardware interrupt causes the CPU 112 to save its state of execution in the memory 114 and begin execution of an interrupt handler. The software interrupt is usually implemented as an instruction in an instruction set, which cause a context switch to an interrupt handler similar to a hardware interrupt.

Commonly used hardware interrupts can be categorized into, but not limited to, a maskable interrupt, a non-maskable interrupt (NMI), an inter-processor interrupt (IPI), and a spurious interrupt. The maskable interrupt, sometimes referred to as an interrupt request (IRQ), is a hardware interrupt that may be ignored by setting a bit in an interrupt mask register's (IMR) bit-mask. The NMI is a hardware interrupt that lacks an associated bit-mask, so that it can never be ignored. The IPI is a special case of interrupt that is generated by one processor to interrupt another processor in a multiprocessor system. The spurious interrupt is a hardware interrupt that is unwanted, and is typically generated by system conditions such as electrical interference on an interrupt line or through incorrectly designed hardware.

In certain embodiments, the NMI can be used to trigger the CPU 112 to enter a safe state. In other words, when the CPU 112 receives an NMI, the CPU 112 enters the safe state. In the safe state, the CPU 112 is powered for a certain period of time to save the critical operational information, and is otherwise non-operational to prevent damage induced by errors. At the end of the certain period of time, the CPU 112 will receive a reset signal to reboot the host computer 110.

The memory 114 can be a volatile memory, such as the random-access memory (RAM), for storing the data and information during the operation of the host computer 110. In certain embodiments, the memory 114 is in communication with the CPU 112 through a system bus (not shown).

The BIOS chip 116 is one of the most crucial components in the host computer 110 or any computer system. In certain embodiments, the BIOS chip 116 is a non-volatile memory, such as a flash memory chip, an electrically erasable programmable read-only memory (EEPROM) chip or a complementary metal oxide semiconductor (CMOS) memory. As shown in FIG. 1, the BIOS chip 116 stores a BIOS image 170 (hereinafter the BIOS 170), an IPMI application 172, and a flasher module 174.

The BIOS 170 is an image file storing the firmware codes which, when executed at the CPU 112, are configured to perform the startup functions, or the booting functions, for the host computer 110. Examples of the booting functions include, but are not limited to, the initiation and power-on self-test, identifying the system devices, locating the boot loader software on the boot device, loading and executing the boot loader software and giving it control of the OS 160 in the protected mode.

As described above, the BIOS 170 can perform the startup or the booting functions. In the booting process, the BIOS 170 performs a power-on self-test operation, which initializes and identifies the system hardware devices of the host computer 110, such as the CPU 112, the memory 114, the storage device 118, peripheral I/O devices such as display card, keyboard and mouse, and other hardware devices. In certain embodiments, the BIOS 170 may identify the storage device 118, which stores an operating system (OS) 160, as a bootable device. Once the BIOS 170 completes the self-test operation, the BIOS 170 then attempts to boot the host computer 110, i.e., instructs the CPU 112 to read and execute the OS 160 from the bootable device, i.e. the storage device 118 as specified in the BIOS 170. Typically, the BIOS 170 attempts to load a boot loader program from the storage device 118. The boot loader program then loads the OS 160 from the storage device 118. Thus, the CPU 112 can execute the OS 160 and run an instance of the OS 160. At the end of the booting process, the control of the computer system is given to the OS 160.

In certain embodiments, after completing the self-test operation without errors, the BIOS 170 may perform a failover backup process if the SP 120 does not have a newer version of the BIOS image 185. In certain embodiments, when the BIOS 170 completes the self-test operation successfully, the BIOS 170 may send a request to the SP 120 asking for version information of the BIOS image 185 stored in the SP 120. In certain embodiments, the version information may be a timestamp showing the time when the BIOS image 185 was generated, or a version identification number indicating the version of the BIOS image 185. When the SP 120 returns the version information, the BIOS 170 compares the version information of the BIOS image 185 with the current version of the BIOS 170 stored in the BIOS chip 116. If the BIOS 170 determines that the BIOS image 185 stored in the SP 120 is an older version than the BIOS 170, the BIOS 170 sends a copy of the BIOS 170 stored in the BIOS chip 116 to the SP 120 as a failover backup image before loading the OS 160. Upon receiving the copy of the BIOS 170, the SP 120 may store the copy of the BIOS 170 in the memory 122 to replace the BIOS image 185. If there is no BIOS image 185 stored in the SP 120, the SP 120 does not return the version information. When the BIOS 170 waits for a certain period of time without receiving the version information from the SP 120, the BIOS 170 determines that there is no BIOS image stored in the SP 120. The BIOS 170 then sends a copy of the BIOS 170 stored in the BIOS chip 116 to the SP 120 as the failover backup image before loading the OS 160. Upon receiving the copy of the BIOS 170, the SP 120 may store the copy of the BIOS 170 in the memory 122 to form the BIOS image 185.

In certain embodiments, the BIOS 170 stored in the BIOS chip 116 may be corruptive. Thus, when the BIOS 170 is executed in the booting process to perform the self-test operation, an error may occur due to the corruption such that the BIOS 170 cannot proceed with loading the OS 160, and the booting will fail. In this case, if the SP 120 has a BIOS image 185 available as a failover backup image, the BIOS 170 may perform a BIOS flashing process by requesting the BIOS image 185 from the SP 120. In certain embodiments, when the BIOS 170 detects that an error occurs, the BIOS 170 may send a request to the SP 120 asking for a copy of the BIOS image 185. Upon receiving the copy of the BIOS image 185, the BIOS 170 loads and executes the flasher module 174 to perform flashing of the BIOS chip 116.

The IPMI application 172 is a computer program that generates and processes IPMI messages. In certain embodiments, when the BIOS 170 is executed at the CPU 112 in the booting process, the BIOS 170 may send data to the IPMI application

11

172 for converting the data between IPMI messages and data recognizable by the BIOS 170. For example, during the booting process, the BIOS 170 may generate a notification to be sent to the SP 120 indicating the condition of the booting process, i.e. successful booting or error occurring. In order to send the notification to the SP 120, the BIOS 170 can send data of the notification to the IPMI application 172. Upon receiving the data of the notification, the IPMI application 172 converts the data to an IPMI OEM message representing the notification, and sends the IPMI OEM message back to the BIOS 170. When the BIOS 170 receive the IPMI OEM messages from the SP 120 or any other peripheral devices, the BIOS 170 can send the received IPMI OEM messages to the IPMI application 172. The IPMI application 172 processes the IPMI OEM messages to generate data recognizable by the BIOS 170, and then sends the data back to the BIOS 170 for further process.

In certain embodiments, the IPMI application 172 is separated from the BIOS 170 as an independent program. In certain embodiments, the IPMI application 172 can be a part of the BIOS 170, which is compatible to the IPMI architecture for generating IPMI messages.

The IPMI messages can be transmitted, translated, bussed, and wrapped in a variety of fashions. Generally, an IPMI message can be a request message (generally labeled as "rq") or a response message (generally labeled as "rs"), and has a plurality of fields, including the network function (NetFn) field, the logical unit number (LUN) field, the sequence number (Seq#) field, the command (CMD) field, and the data field.

The NetFn field is a six-bit field that describes the network function of the IPMI message. The IPMI Specification defines certain predefined categories for the NetFn field, and there are reserved codes for OEM expansion. For a request message, the NetFn field is an odd number, and for a response message, the NetFn field is an even number.

The LUN field is a two-digit logical unit number for sub-addressing purposes. For low-end integrity servers and computers, the LUN field is always zero.

The Seq# field defines a sequence number in the response message to identify the response message in response to a certain request message. In certain embodiments, the value stored in the Seq# field is the same number as the sequence number appeared in the CMD field of the request message to which the response message is in response. As discussed above, the Seq# field must be provided when the NetFn field is an even number to indicate the IPMI message as a response message.

The CMD field includes the commands of the IPMI message. When the IPMI message is a request message, the CMD field includes a sequence number such that a response message to the request message may include the same sequence number in the Seq# field.

The data field includes all data transferred by the IPMI message. In certain embodiments, the data field of an IPMI message can be 0 bytes. In other words, an IPMI message may contain no data field.

FIG. 2 schematically depicts an IPMI message according to certain embodiments of the present disclosure. As shown in FIG. 2, each block of the IPMI message 200 refers to a field, which may have a different length. The IPMI message 200 includes a plurality of fields, such as the rs slave address (rsSA) field 201, the NetFn field 202, the rq slave address (rqSA) field 204, the rq Seq# (rqSeq)/rqLUN field 205, the command field 206, and the data field 207. Further, a plurality of checksum fields 203 and 208 are included for checksum

12

verification purposes. As discussed above, the data field 207 may contain 0 bytes of data (i.e. no data field) or more than one byte of data.

In certain embodiments, IPMI OEM messages are used for data transaction purposes between the BIOS 170 and the SP 120. When the IPMI application 172 processes data to be transferred to the SP 120 to generate an IPMI OEM message, the data can be stored in the data field 207 of the IPMI OEM message as shown in FIG. 2.

The flasher module 174 is a program to perform flashing of the BIOS image 170 stored in the BIOS chip 116. In certain embodiments, flashing of the BIOS 170 can be performed due to a newer BIOS image 185 available in the SP 120, or due to an error occurring during the booting process. When the BIOS 170 receives a BIOS image for flashing the BIOS chip 116, the BIOS 170 calls the flasher module 174 to perform the flashing process, the flasher module 174 receives instructions from the BIOS 170 and performs flashing functions to flash the BIOS chip 116. In certain embodiments, the flasher module 174 can be a part of the BIOS 170. In certain embodiments, the flasher module 174 can be an independent program separated from the BIOS 170.

In certain embodiments, the flasher module 174 may be stored in a storage media other than the BIOS chip 116. In certain embodiments, the flasher module 174 may be stored in the storage device 118 of the host computer 110. In certain embodiments, the flasher module 174 may be stored as a part of the firmware 126 in the SP 120 such that the SP 120, instead of the BIOS 170, may execute the flasher module 174.

As discussed above, the BIOS chip 116 is a non-volatile memory such as a flash memory chip. Typically, the flash memory stores information in an array of memory cells made from floating-gate transistors, which is different in its nature from other volatile or non-volatile memory because the information or data stored therein must be erased before new data can be written to the memory cells. There are two main types of flash memory: the NAND type and the NOR type, which are respectively named after the NAND and NOR logic gates. The flash memory is divided into logical blocks. Each block can vary in size, where the most common is 128 KB. In the majority of NAND flash devices each block is made of 64 pages of 2 KB each. A page is divided in two regions: the data area, and the spare area used for memory management purposes. Pages are divided in sector units (or chunks) of 512 byte to emulate the popular sector size (ibid). The block is the smallest erasable unit while the page is the smallest programmable unit.

FIG. 3 schematically depicts the BIOS chip storing the BIOS according to certain embodiments of the present disclosure. As shown in FIG. 3, the BIOS chip 116 includes a plurality of logical blocks 210, and data of the BIOS 170 is stored in a number of the blocks 210. In certain embodiments, data of the BIOS 170 may be distributed between the blocks 210. In other words, each block 210 may include some data of the BIOS 170 and some empty memory space. It should be appreciated that FIG. 3 shows the blocks 210 in the matrix form solely for the illustration purposes, and the actual memory allocation of the blocks 210 of the BIOS chip 116 may be different.

In certain embodiments, the flasher module 174 utilizes a flash driver (or flash translation layer) to read and write data to the BIOS chip 116. For example, under a Linux system, the executed flasher module 174 generally accesses the flash memory of the BIOS chip 116 through the memory technology device (MTD) subsystem. In certain embodiments, the executed flasher module 174 can mount a file system on top of the MTD subsystem. The flash driver of the MTD subsystem

13

operates a block **210** as the smallest erasable unit. In certain embodiments, a block **210** can have a size of 128K (=131072) bytes. In an erasing or rewriting operation, data in one block **210** must be erased before new data can be rewritten to any sector of the block **210**. When a file system is mounted on top of the MTD subsystem, the file system uses sectors (not shown in FIG. 3A) as the basic memory units. The size of a sector is generally 512 or 1024 bytes. A block contains a number of sectors. Data can be written into one or more sectors of a block. Examples of the file system include ext2, ext3, XFS, JFS, FAT, or any other suitable file systems.

FIG. 4 schematically depicts a flashing process of the BIOS chip according to certain embodiments of the present disclosure. In certain embodiments, the BIOS **170** has saved a copy of replacement BIOS image in the memory **112**, and executes the flasher module **174** to perform the flashing process.

At procedure **410**, when the flashing process starts, the flasher module **174** compares one block **210** of the BIOS chip **116** with a corresponding block of the BIOS image stored in the memory **112** to determine whether the two blocks are the same (e.g., include the same content). At procedure **420**, if the flasher module **174** determines that the two blocks are different, the flasher module **174** enters procedure **430**. If the two blocks are the same, the flasher module **174** enters procedure **440**. At procedure **430**, the flasher module **174** erases the block **210** of the BIOS chip **116**, and writes the corresponding block of the replacement BIOS image to the erased block **210**. At procedure **440**, the flasher module **174** skips the block **210**. In other words, the block **210** of the BIOS chip **116** is not changed if the contents stored in the block **210** are the same to the corresponding block of the BIOS image stored in the memory **112**.

After completing a block **210** of the BIOS chip **116**, at procedure **450**, the flasher module **174** checks if that just compared block **210** is the last block of the BIOS chip **116**. If the block **210** is the last block, the flashing process is completed. If there are other blocks waiting to be operated, the flasher module **174** enters procedure **460** to go back to procedure **410** to operate on the next block **210**. In this way, the flasher module **174** processes through all the blocks **210** of the BIOS chip **116**. When the flashing process is completed, the flasher module **174** may return control to the BIOS **170**, and the BIOS **170** may restart the host computer **110**.

The storage device **118** is a non-volatile data storage media for storing the OS **160**, the IPMI application **172**, and other applications of the host computer **110**. Examples of the storage device **118** may include flash memory, memory cards, USB drives, hard drives, floppy disks, optical drives, or any other types of data storage devices.

The OS **160** can be collective management software managing the operation of the host computer **110**. For example, the OS **160** can include a set of functional programs that control and manage operations of the devices connected to the host computer **110**. The set of application programs provide certain utility software for the user to manage the host computer **110**. In certain embodiments, the OS **160** is operable to multitask, i.e., execute computing tasks in multiple threads, and thus may be any of the following: MICROSOFT CORPORATION's "WINDOWS 95," "WINDOWS CE," "WINDOWS 98," "WINDOWS 2000" or "WINDOWS NT", "WINDOWS Vista," "WINDOWS 7," and "WINDOWS 8," operating systems, IBM's OS/2 WARP, APPLE's MACINTOSH OSX operating system, LINUX, UNIX, etc. In certain embodiments, the OS **160** can also be compatible to the IPMI architecture for generating IPMI messages in order to communicate with the SP **120**.

14

The SP **120** is a specialized microcontroller that manages the interface between system management software and platform hardware. In certain embodiments, the SP **120** may be a baseboard management controller (BMC). Different types of sensors can be built into the host computer **110**, and the SP **120** reads these sensors to obtain parameters such as temperature, cooling fan speeds, power status, OS status, etc.

The SP **120** monitors the sensors and can send out-of-band (OOB) alerts to a system administrator of the host computer **110** if any of the parameters do not stay within preset limits, indicating a potential failure of the host computer **110**. In certain embodiments, the administrator can also remotely communicate with the SP **120** from a remote management computer via a network to take remote action to the host computer. For example, the administrator may reset the host computer **110** from the remote management computer through the SP **120**, and may obtain system information of the host computer **110** OOB without interrupting the operation of the host computer **110**.

As shown in FIG. 1, the SP **120** includes a processor **121**, a memory **122** and a non-volatile memory **124**. In certain embodiments, the SP **120** is also connected to the BIOS chip **116** of the host computer **110** directly through a SPI **158** such that the SP **120** has direct SPI access to the BIOS chip **116** through the SPI **158**. In certain embodiments, the SP **120** may include other components, such as at least one I/O device (not shown).

The processor **121** controls operation of the SP **120**. The processor **121** can execute the firmware **126** or other codes stored in the SP **120**. In certain embodiments, the SP **120** may run on or more than one processor.

The memory **122** can be a volatile memory, such as the RAM, for storing the data and information during the operation of the SP **120**. When the SP **120** restarts, the contents stored in the memory **122** will be lost.

In certain embodiments, the memory **122** is configured to store a copy of BIOS image **185**. The BIOS image **185** is a copy of BIOS image file storing in the memory **122** of the SP **120**. In certain embodiments, the BIOS image **185** contains the version information of the BIOS image **185**. In certain embodiments, the version information may be a timestamp showing the time when the BIOS image **185** was generated, or a version identification number indicating the version of the BIOS image **185**.

In certain embodiments, the SP **120** receives the copy of the BIOS image **185** as a failover backup image from the host computer **110**, and store the BIOS image in the memory **122**.

The non-volatile memory **124** stores the firmware **126** of the SP **120**. The firmware **126** of the SP **120** includes computer executable codes for performing the operation of the SP **120**. As shown in FIG. 1, the firmware **126** of the SP **120** includes, among other things, a BIOS update module **180**, a watchdog module **182**, and an IPMI module **190**. In certain embodiments, the firmware **126** may include a web connection module (not shown) for communication with the network such that the administrator of the computer system may connect to the SP **120** remotely from a remote management computer via the network.

The BIOS update module **180** controls the transmission of the BIOS image between the SP **120** and the host computer **110** via the system interface, such as the KCS interface **150** or the USB interface **155**. In certain embodiments, when the BIOS **170** sends a request for the version information of the BIOS image **185** to the SP **120**, the BIOS update module **180** may retrieve version information of the BIOS image **185** from the memory **122**, and send version information of the BIOS image **185** to the firmware **126** such that the firmware **126**

15

may send the version information to the BIOS 170 at the host computer 110 for comparing of the version of the BIOS images. In certain embodiments, when the BIOS 170 sends a request to the BIOS update module 180 for the BIOS image 185, the BIOS update module 180, in response to the request, retrieves a copy of the BIOS image 185 from the memory 122, and sends the copy of the BIOS image 185 to the firmware 126 such that the firmware 126 may send the copy of the BIOS image 185 to the BIOS chip 116 to replace the BIOS 170 stores in the BIOS chip 116. In certain embodiments, when the BIOS 170 sends a copy of the BIOS 170 to the SP 120, the firmware 126 receives the copy of the BIOS 170 and sends the copy of the BIOS 170 to the BIOS update module 180. The BIOS update module 180 then saves the copy of the BIOS 170 in the memory 122 to replace or to form the BIOS image 185.

In certain embodiments, the error occurs in the booting process may be critical that the BIOS 170 stops operating with no capability of retrieving the copy of the BIOS image 185 from the SP 120 to flash the BIOS chip 112. In this case, if the SP 120 is connected to the BIOS chip 116 of the host computer 110 through the SPI 158, the watchdog module 182 may trigger the CPU 112 of the host computer 110 to enter the safe state, and then instruct the BIOS update module 180 to perform flashing of the BIOS 170 directly through the SPI 158. In this case, the BIOS update module 180 provides similar functionalities of the flasher module 174 at the host computer 110. The flashing process performed by the BIOS update module 180 is described with reference to FIG. 4, and is hereinafter not repeated.

The watchdog module 182 is a monitoring program to detect and recover malfunctions of the host computer 110. As discussed above, the host computer 110 may have different types of sensors built therein, and the watchdog module 182 of the SP 120 may read these sensors to obtain parameters and conditions of the operation of the host computer 110. In certain embodiments, the watchdog module 182 may be used to monitor the operation of the BIOS 170 in the booting process of the host computer 110. When an error occurs in the booting process, the watchdog module 182 may send alerts to the administrator of the system indicating a potential failure of the system.

In certain embodiments, the watchdog module 182 can be an electronic timer program with a predetermined period of time. The watchdog timer constantly receives notification signals from the host computer 110 to restart the watchdog timer in order to prevent the timer from elapsing or timing out. The act of restarting a watchdog timer of the watchdog module 182 is sometimes referred to as “kicking the dog,” and the notification signal to restart the watchdog timer may be referred to as a “kick” signal. For example, during normal operation of the booting process, the BIOS 170 may regularly send kick signals to the SP 120 to restart the watchdog timer of the watchdog module 182. If an error occurs during the booting process, the BIOS 170 stops processing with the follow-up booting procedures. Thus, the BIOS 170 will not continue sending the kick signal to the SP 120 to restart the watchdog timer of the watchdog module 182. When the watchdog timer elapses, the watchdog module 182 may generate a timeout signal to initiate certain corrective actions to the error of the BIOS 110.

In certain embodiments, the watchdog module 182 can have a variety of configurations, which is subject to altering. For example, the watchdog module 182 can be one or more timers, and each timer of the watchdog module 182 is generally referred to as a stage. In other words, a single-stage watchdog module has one single timer, and a multi-stage watchdog module has multiple timers. For a multi-stage

16

watchdog module, each stage or each timer may have a different predetermined period of time.

FIG. 5 schematically depicts operation of a watchdog module according to certain embodiments of the present disclosure. As shown in FIG. 5, the watchdog module 182 is a single-stage watchdog program having a timer 186. In certain embodiments, the watchdog module 182 can be a multiple-stage watchdog program having a plurality of timers.

During the regular operation of the booting process, the BIOS 170 controls the CPU 112 to regularly send the kick signal to the SP 120. Specifically, the kick signal can be sent in the IPMI OEM message format to the SP 120 through the KCS interface 150. It should be appreciated that the IPMI OEM message can be sent to the SP 120 through the USB interface 155 or any other system interfaces.

Upon receiving the kick signal within the predetermined period of time, the timer 186 of the watchdog module 182 restarts without sending out any timeout signals. When an error occurs during the booting process, the BIOS 170 stops processing, and the CPU 112 stops sending the kick signal to the SP 120, causing the timer 186 to elapse. When the timer 186 elapses, the timer 186 sends out a timeout signal to the BIOS update module 180 and to the CPU 112. In certain embodiments, the timeout signal can be a NMI. Specifically, the watchdog module 182 may send the NMI timeout signal as a general purpose input/output (GPIO) signal to the CPU 112 of the host computer 110.

When the CPU 112 receives the NMI timeout signal, the CPU 112 enters the safe state. At the same time, the watchdog module 182 sends a timeout signal to invoke the BIOS update module 180. In certain embodiments, the timeout signal to invoke the BIOS update module 180 can be merely a timeout signal and not a NMI. Upon receiving the timeout signal, the BIOS update module 180 starts the flashing process of the BIOS chip 116 of the host computer 110 through the SPI 158. In certain embodiments, the BIOS update module 180 may wait for a certain period of time to give the CPU 112 time to save the critical operational information, such as writing an error message to a log file.

When the BIOS update module 180 completes the flashing process, the BIOS update module 180 sends a restart signal to the CPU 112 to restart the host computer 110. Specifically, the BIOS update module 180 may send the restart signal as a GPIO signal to the CPU 112 of the host computer 110.

The IPMI module 190 is a program of the SP 120 to generate and process IPMI messages. In certain embodiments, when the firmware 126 of the SP 120 generates data to be transferred to the host computer 110 under the IPMI architecture, the firmware 126 sends the data to the IPMI module 190. Upon receiving the data, the IPMI module 190 converts the data to corresponding IPMI OEM messages, and sends the IPMI OEM messages back to the firmware 126 such that the firmware 126 may send the IPMI OEM messages to the host computer 110. When the firmware 126 receives IPMI OEM messages from the host computer 110 or any other IPMI compatible devices, the firmware 126 sends the received IPMI OEM messages to the IPMI module 190. The IPMI module 190 processes the IPMI OEM messages to generate data recognizable by the firmware 126, and then sends the data back to the firmware 126 for further process. In certain embodiments, the IPMI module 190 can be a similar program to the IPMI application 172 at the host computer 110.

Currently, almost all firmware of SP's or BMC's available in the market supports the IPMI architecture, and provide a variety of IPMI modules 190. In certain embodiments, the IPMI module 190 can be a part of the firmware 126, which is compatible to the IPMI architecture for generating IPMI mes-

17

sages. In certain embodiments, the IPMI module **190** is separated from the firmware **126** as an independent program.

In certain embodiments, IPMI OEM messages are used for data transaction purposes between the BIOS **170** and the SP **120**. When the IPMI module **190** processes data to be transferred to the host computer **110** to generate an IPMI OEM message, the data can be stored in the data field **447** of the IPMI OEM message as shown in FIG. **2**.

The KCS interface **150** is a standardized interface often used between a SP and a payload processor in the IPMI architecture. IPMI is an industry standard for system monitoring and event recovery. The IPMI specification provides a common message-based interface for accessing all of the manageable features in a compatible computer. IPMI includes a rich set of predefined commands for reading temperature, voltage, fan speed, chassis intrusion, and other parameters. System event logs, hardware watchdogs, and power control can also be accessed through IPMI. In this manner, IPMI defines protocols for accessing the various parameters collected by a SP through an operating system or through an external connection, such as through a network or serial connection. Additional details regarding IPMI can be found in the IPMI Specification (Version 2.0), which is publicly available from INTEL CORPORATION, and which is incorporated herein by reference.

In certain embodiments, in addition to the standard predefined commands and parameters, IPMI allows OEM extensions for the manufacturers and users to define OEM specific commands. A user may use IPMI OEM messages to control data transmission via the KCS interface **150**. The IPMI OEM messages may be used for the data transaction between the host computer **110** and the SP **120**. In certain embodiments, the IPMI OEM messages may include the specific data to be transferred, or the request for the specific data.

The USB interface **155** is an industry standardized interface under the USB industry standard that defines the cables, connectors and communications protocols used in a bus for connection, communication, and power supply between computers and electronic devices. In certain embodiments, the USB interface **155** is a USB port.

USB has effectively replaced a variety of earlier interfaces, such as serial and parallel ports, as well as separate power chargers for portable devices. Currently, USB allows bi-directional communications between the host computer **110** and the SP **120**, as USB 3.0 allows for device-initiated communications towards the host.

In certain embodiments, the USB interface **155** may be used to transfer IPMI OEM messages between the host computer **110** and the SP **120**. In certain embodiments, when the SP **120** is connected to the host computer **110** via the USB interface **155**, the OS **160** may request and receive a plurality of USB descriptors from the SP **120** through the USB interface **130**. Based on information of the USB descriptors, the OS **160** may recognize the BMC **120** as the specific USB human interface device (HID) device, and recognize a predefined format of a USB HID report for transferring data to the specific HID device (i.e. the BMC **120**). In certain embodiments, the USB HID report is in the predefined format based on the information of the USB descriptors with the IPMI OEM messages embedded therein. Thus, the IPMI OEM messages can be transferred between the host computer **110** and the SP **120** through the USB interface **155** by transferring the USB HID reports with the IPMI OEM messages embedded therein.

In certain embodiments, the SP **120** receives the copy of the BIOS image **185** as a failover backup image from the host computer **110**, and store the BIOS image in the memory **122**.

18

FIG. **6** schematically depicts a SP receiving a failover backup BIOS image from the BIOS of the host computer in the booting process according to certain embodiments of the present disclosure. At procedure **610**, the host computer **110** starts booting. At procedure **612**, the CPU **112** loads a copy of the BIOS **170** from the BIOS chip **116**. At procedure **614**, the CPU **112** saves the copy of the BIOS **170** in the memory **114** and executes the BIOS **170**. At procedure **616**, the BIOS **170**, when executed at the CPU **112**, performs the booting process, such as the power-on self-test operation for initializing and identifying the system hardware devices of the host computer **110**.

When the BIOS **170** finishes the booting process without generating any error, the BIOS **170** may send a copy of the BIOS image **170** stored in the BIOS chip **116** to the SP **120** as a failover backup image. At procedure **620**, the BIOS **170** finishes the booting process without generating any error. At procedure **622**, the BIOS **170** sends a copy of the BIOS image **170** to the IPMI application **172** for processing. Upon receiving the copy of the BIOS image **170**, at procedure **625**, the IPMI application **172** converts the copy of the BIOS image **170** to generate the IPMI OEM message. At procedure **628**, the IPMI application **172** sends the IPMI OEM message back to the BIOS **170**.

At procedure **630**, the BIOS **170** sends the IPMI OEM message representing the copy of the BIOS image **170** to the SP **120** through the KCS interface **150**. It should be appreciated that the IPMI OEM message can be sent to the SP **120** through the USB interface **155** or any other system interfaces.

When the SP **120** receives the IPMI OEM message representing the copy of the BIOS image **170**, at procedure **632**, the firmware **126** of the SP **120** sends the IPMI OEM message to the IPMI module **190** for processing. Upon receiving the IPMI OEM message, at procedure **635**, the IPMI module **190** converts the IPMI OEM message to generate the copy of the BIOS image **170**, which is recognizable by the firmware **126**. At procedure **638**, the IPMI module **190** sends the copy of the BIOS image **170** back to the firmware **126**.

Upon receiving the copy of the BIOS image **170**, at procedure **640**, the firmware **126** calls the BIOS update module **180**. At procedure **650**, the BIOS update module **180** stores the received copy of the BIOS image **170** in the memory **122** to form the BIOS image **185** as a failover backup image. In certain embodiments, when the memory **122** already has a BIOS image **185** stored therein, the BIOS update module **180** replaces the BIOS image **185** with the received BIOS image file **145**. In certain embodiments, the BIOS image **185** contains the version information.

In certain embodiments, the SP **120** has the capability to direct access to the SPI **158** and directly connect to the BIOS chip **116** of the host computer **110**. Further, during a previous booting process of the host computer **110**, the SP **120** has stored a copy of the BIOS image **185** in the memory **122** as a failover backup image from the host computer **110**. In this case, the SP **120** may monitor the booting process of the host computer **110** with the watchdog module **182**. When the watchdog module **182** detects an error in the booting process of the host computer **110**, the SP **120** may initiate a flashing process of the BIOS chip **116** directly through the SPI **158**. For example, the BIOS **170** in the booting process may encounter a major error due to corruption of the BIOS **170** in the BIOS chip **116**. The error causes the BIOS **170** to fail the booting process and to lose its interaction with the SP **120**. In this case, the BIOS **170** cannot initiate the flashing process. Thus, the watchdog module **182** of the SP **120** may initiate a flashing process through the SPI **158** in order to replace the corruptive BIOS image **170** in the BIOS chip **116**.

19

FIG. 7 schematically depicts a BIOS failover flashing process initiated by the SP when the SP has direct SPI access to the BIOS chip according to certain embodiments of the present disclosure. In certain embodiments, the BIOS image 185 stored in the SP 120 is a failover backup image obtained from the host computer 110 during a previous booting process, as described with reference to FIG. 7. In certain embodiments, the watchdog module 182 of the SP 120 is a single-stage watchdog program as described with reference to FIG. 5.

As described above, the watchdog module 182 monitors the operation of the booting process at the host computer 110. The monitoring process of the watchdog module 182 is performed by receiving the kick signals constantly from the BIOS 170 of the host computer 110, which has been described with reference to FIG. 5. If an error occurs during the booting process, the BIOS 170 stops processing with the follow-up booting procedures. Thus, the BIOS 170 will not continue sending the kick signal to the SP 120 to restart the watchdog timer of the watchdog module 182. At procedure 710, when the timer 186 of the watchdog module 182 elapses, the timer 186 generates a timeout signal for the CPU 112 and the BIOS update module 180.

At procedure 720, the watchdog module 182 may send the timeout signal as a general purpose input/output (GPIO) signal to the CPU 112 of the host computer 110 to trigger the CPU 112 to enter the safe state.

At procedure 730, the watchdog module 182 sends the timeout signal to the BIOS update module 180 to initiate the flashing process. It should be appreciated that, although FIG. 7 shows that the watchdog module 182 sends the timeout signal to the CPU 112 (procedure 720) and to the BIOS update module 180 (procedure 730) in separate procedures, the watchdog module 182 sends out the timeout signal simultaneously to the CPU 112 and to the BIOS update module 180. In other words, the CPU 112 and to the BIOS update module 180 may process simultaneously. In certain embodiments, before initiating the flashing process, the BIOS update module 180 may wait for a certain period of time, e.g., 5 seconds, to give the CPU 112 time to save the critical operational information, such as writing an error message to a log file.

At procedure 740, in response to the timeout signal, the BIOS update module 180 checks the BIOS image 185 stored in the memory 122, and at procedure 745, the BIOS update module 180 retrieves a copy of the BIOS image 185 stored in the memory 122. At procedure 750, the BIOS update module 180 sends the copy of the BIOS image 185 back to the firmware 126.

At procedure 760, the firmware 126 instructs the BIOS update module 180 to perform flashing of the BIOS chip 116 with the BIOS image 185 through the direct access of SPI 158. The process of flashing the BIOS chip 116 has been described with reference to FIG. 4, and is hereinafter not repeated.

When the flashing process is complete, the BIOS update module 180 returns control to the firmware 126, and at procedure 770, the firmware 126 generates a restart signal to restart the host computer 110. At procedure 780, the firmware 126 sends the restart signal to the CPU 112 to trigger the CPU 112 to reboot with the new BIOS image 170 in the BIOS chip 116. At procedure 790, the CPU 112 restarts the host computer 110.

The foregoing description of the exemplary embodiments of the disclosure has been presented only for the purposes of illustration and description and is not intended to be exhaus-

20

tive or to limit the disclosure to the precise forms disclosed. Many modifications and variations are possible in light of the above teaching.

The embodiments were chosen and described in order to explain the principles of the disclosure and their practical application so as to enable others skilled in the art to utilize the disclosure and various embodiments and with various modifications as are suited to the particular use contemplated. Alternative embodiments will become apparent to those skilled in the art to which the present disclosure pertains without departing from its spirit and scope. Accordingly, the scope of the present disclosure is defined by the appended claims rather than the foregoing description and the exemplary embodiments described therein.

What is claimed is:

1. A system, comprising:

a service processor (SP), comprising a processor, a non-volatile memory, a volatile memory and a system interface, wherein the non-volatile memory stores computer executable codes are configured to, when executed at the processor,

receive, from a basic input/output system (BIOS) being executed at a central processing unit (CPU) of a host computer through a system interface, a BIOS image as a failover backup image, and store the failover backup image in the volatile memory, wherein the SP is connected to the host computer via the system interface, and wherein the SP is connected to a BIOS chip of the host computer via a serial peripheral interface (SPI);

monitor operation of the BIOS being executed at the CPU of the host computer by receiving, from the BIOS through the system interface, a notification signal;

detect, based on the notification signal, an error in the operation of the BIOS being executed at the CPU of the host computer; and

in response to detecting the error, send a copy of the failover backup image to the BIOS chip of the host computer through the SPI to replace a current BIOS image stored in the BIOS chip of the host computer with the copy of the failover backup image.

2. The system as claimed in claim 1, wherein the system interface is a standardized interface under an Intelligent Platform Management Interface (IPMI) standard, wherein the standardized interface comprises a keyboard controller style (KCS) interface, a system management interface chip (SMIC) interface, and a block transfer (BT) interface.

3. The system as claimed in claim 1, wherein the system interface is a universal serial bus (USB) interface.

4. The system as claimed in claim 1, wherein the SP is a baseboard management controller (BMC).

5. The system as claimed in claim 1, wherein the codes comprise:

a detection module configured to receive the notification signal from the BIOS being executed at the CPU of the host computer through the system interface, and to detect the error based on the notification signal.

6. The system as claimed in claim 5, wherein the detection module is configured to detect the error when the detection module does not receive the notification signal for a predetermined period of time.

7. The system as claimed in claim 1, wherein the host computer comprises:

the BIOS chip storing the current BIOS image; a memory; and

21

the CPU, configured to load a copy of the current BIOS image to the memory and execute the current BIOS image as the BIOS;

wherein the BIOS, when executed at the CPU, is configured to

- retrieve a copy of the current BIOS image from the BIOS chip, and send the copy of the current BIOS image as the failover backup image to the SP through the system interface; and
- send the notification signal to the SP during the operation of the BIOS.

8. The system as claimed in claim 7, wherein the BIOS, when executed at the CPU, is configured to send the copy of the current BIOS image as the failover backup image to the SP by:

- generating a first IPMI original equipment manufacturer (OEM) message comprising the copy of the current BIOS image; and
- sending the first IPMI OEM message to the SP through the system interface.

9. The system as claimed in claim 8, wherein the codes are configured to receive the BIOS image as the failover backup image from the BIOS being executed at the CPU of the host computer by:

- receiving the first IPMI OEM message from the BIOS being executed at the CPU of the host computer through the system interface; and
- retrieving the BIOS image from the first IPMI OEM message.

10. The system as claimed in claim 7, wherein the notification signal is a second IPMI OEM message.

11. The system as claimed in claim 7, wherein the BIOS, when executed at the CPU, is further configured to:

- generate a request for version information of the failover backup image, and send the request for the version information of the failover backup image to the SP through the system interface;
- receive, from the SP through the system interface, the version information of the failover backup image; and
- compare the version information of the failover backup image to a version information of the current BIOS image.

12. The system as claimed in claim 11, wherein the codes are configured to:

- receive, from the BIOS being executed at the CPU of the host computer through the system interface, the request for the version information of the failover backup image;
- in response to the request for the version information, retrieve the version information from the failover backup image stored in the volatile memory; and
- send the version information to the host computer through the system interface.

13. The system as claimed in claim 11, wherein the request for the version information of the failover backup image is a third IPMI OEM message, and the version information of the failover backup image is a fourth IPMI OEM message.

14. A method of failover updating a basic input/output system (BIOS) of a host computer with a service processor (SP), comprising:

- receiving, at the SP, a failover backup image from the host computer through a system interface, and storing the failover backup image in a volatile memory of the SP, wherein the host computer comprises a central processing unit (CPU), a memory and a BIOS chip storing a current BIOS image, wherein the CPU is configured to load a copy of the current BIOS image to the memory and execute the current BIOS image as a BIOS, wherein

22

the BIOS being executed at the CPU of the host computer is configured to retrieve a copy of the current BIOS image from the BIOS chip and send the copy of the current BIOS image as the failover backup image to the SP through the system interface, and wherein the SP is connected to the BIOS chip of the host computer via a serial peripheral interface (SPI);

- monitoring, at the SP, operation of the BIOS being executed at the CPU of the host computer by receiving, from the BIOS through the system interface, a notification signal;
- detecting, at the SP, an error in the operation of the BIOS being executed at the CPU of the host computer based on the notification signal; and
- in response to detecting the error, sending, by the SP, a copy of the failover backup image to the BIOS chip of the host computer through the SPI to replace the current BIOS image stored in the BIOS chip of the host computer with the copy of the failover backup image.

15. The method as claimed in claim 14, wherein the system interface is a standardized interface under an Intelligent Platform Management Interface (IPMI) standard, wherein the standardized interface comprises a keyboard controller style (KCS) interface, a system management interface chip (SMIC) interface, and a block transfer (BT) interface.

16. The method as claimed in claim 14, wherein the system interface is a universal serial bus (USB) interface.

17. The method as claimed in claim 14, wherein the SP is a baseboard management controller (BMC).

18. The method as claimed in claim 14, wherein the SP comprises a detection module configured to receive the notification signal from the BIOS being executed at the CPU of the host computer through the system interface, and to detect the error based on the notification signal.

19. The method as claimed in claim 18, wherein the detection module is configured to detect the error when the detection module does not receive the notification signal for a predetermined period of time.

20. The method as claimed in claim 14, wherein the SP is configured to receive the failover backup image from the BIOS being executed at the CPU of the host computer by:

- receiving a first IPMI original equipment manufacturer (OEM) message from the BIOS being executed at the CPU of the host computer through the system interface, wherein the first IPMI OEM message is generated by the BIOS being executed at the CPU of the host computer and comprises the copy of the current BIOS image; and
- retrieving the copy of the current BIOS image as the failover backup image from the first IPMI OEM message;

wherein the notification signal is a second IPMI OEM message.

21. The method as claimed in claim 14, further comprising:

- receiving, at the SP, a request for the version information of the failover backup image from the BIOS being executed at the CPU of the host computer through the system interface;
- in response to the request for the version information, retrieving, at the SP, the version information from the failover backup image stored in the volatile memory; and
- sending, from the SP, the version information to the host computer through the system interface, wherein the BIOS being executed at the CPU of the host computer receives and compares the version information of the failover backup image to a version information of the current BIOS image;

23

wherein the request for the version information of the failover backup image is a third IPMI OEM message, and the version information of the failover backup image is a fourth IPMI OEM message.

22. A non-transitory computer readable medium storing computer executable codes, wherein the codes, when executed at a processor of a service processor (SP), are configured to:

receive a failover backup image from a host computer through a system interface, and storing the failover backup image in a volatile memory of the SP, wherein the host computer comprises a central processing unit (CPU), a memory and a BIOS chip storing a current BIOS image, wherein the CPU is configured to load a copy of the current BIOS image to the memory and execute the current BIOS image as a BIOS, wherein the BIOS being executed at the CPU of the host computer is configured to retrieve a copy of the current BIOS image from the BIOS chip and send the copy of the current BIOS image as the failover backup image to the SP through the system interface, and wherein the SP is connected to the BIOS chip of the host computer via a serial peripheral interface (SPI);

monitor operation of the BIOS being executed at the CPU of the host computer by receiving, from the BIOS through the system interface, a notification signal;

detect an error in the operation of the BIOS being executed at the CPU of the host computer based on the notification signal; and

in response to detecting the error, send a copy of the failover backup image to the BIOS chip of the host computer through the SPI to replace the current BIOS image stored in the BIOS chip of the host computer with the copy of the failover backup image.

23. The non-transitory computer readable medium as claimed in claim 22, wherein the system interface is a standardized interface under an Intelligent Platform Management Interface (IPMI) standard, wherein the standardized interface comprises a keyboard controller style (KCS) interface, a system management interface chip (SMIC) interface, and a block transfer (BT) interface.

24. The non-transitory computer readable medium as claimed in claim 22, wherein the system interface is a universal serial bus (USB) interface.

25. The non-transitory computer readable medium as claimed in claim 22, wherein the SP is a baseboard management controller (BMC).

24

26. The non-transitory computer readable medium as claimed in claim 22, wherein the codes comprise a detection module configured to receive the notification signal from the BIOS being executed at the CPU of the host computer through the system interface, and to detect the error based on the notification signal.

27. The non-transitory computer readable medium as claimed in claim 26, wherein the detection module is configured to detect the error when the detection module does not receive the notification signal for a predetermined period of time.

28. The non-transitory computer readable medium as claimed in claim 22, wherein the codes are configured to receive the failover backup image from the BIOS being executed at the CPU of the host computer by:

receiving a first IPMI original equipment manufacturer (OEM) message from the BIOS being executed at the CPU of the host computer through the system interface, wherein the first IPMI OEM message is generated by the BIOS being executed at the CPU of the host computer and comprises the copy of the current BIOS image; and retrieving the copy of the current BIOS image as the failover backup image from the first IPMI OEM message;

wherein the notification signal is a second IPMI OEM message.

29. The non-transitory computer readable medium as claimed in claim 22, wherein the codes are further configured to:

receive a request for the version information of the failover backup image from the BIOS being executed at the CPU of the host computer through the system interface;

in response to the request for the version information, retrieve the version information from the failover backup image stored in the volatile memory; and

send the version information to the host computer through the system interface, wherein the BIOS being executed at the CPU of the host computer receives and compares the version information of the failover backup image to a version information of the current BIOS image;

wherein the request for the version information of the failover backup image is a third IPMI OEM message, and the version information of the failover backup image is a fourth IPMI OEM message.

* * * * *